

REMARKS

Claims 1-22 are pending. All claims were rejected under 35 U.S.C. § 102(b) and/or 35 U.S.C. § 103(a). These rejections are traversed.

Rejections under 35 U.S.C. § 102(b)

Claims 1, 3, 4, 9-10, 12-13, 18-20 and 22 were rejected under 35 U.S.C. § 102(b) based on U.S. Patent No. 5,732,271 to Berry. To establish a prima facie case for anticipation under 35 U.S.C. § 102(b), the cited reference must teach every aspect of the claimed invention either explicitly or impliedly. For the reasons discussed below, it is respectfully submitted that the Examiner has not established a prima facie case under 35 U.S.C. § 102(b) for any of the present claims, and that therefore, the present claims are allowable.

The invention generally relates to a technique for optimizing the handling of changes to option values. The present invention provides an option data structure and a mapping data structure that enable option-change handling code to be identified efficiently when an option-change notification occurs. An object is defined with an option data structure, which supports references to options without having to preallocate memory for the full option values. The option data structure can be, for example, a linked list. The invention uses a mapping data structure, which preferably maps an option name and class to an option binding. The mapping data structure can be, for example, a hash table. The option binding identifies a change handler that notifies objects that are affected by a change in an option value.

With the option data structure of the present invention, the potential option values for an object do not need to be physically stored on the object. This can result in substantial savings of memory space, when there is a very large set of possible full option values associated with the object or class that are seldom set on the object. Further, by precomputing mappings of option names and class to an option binding, a change in an object's option value can be effected seamlessly, without having to iterate through the object's or class's property names to determine which property is affected by the change. Thus, the use of the claimed option and mapping data

structures makes it possible to define a very large set of possible options for a class without the time and storage space limitations associated with the prior art.

Conversely, Berry's system provides a mechanism for defining a derived object that is based on a prototypical object, where property values of the derived object can be defined directly on the derived object or can be inherited from the prototypical object. In Berry's system, a change in a property value of a prototypical object causes derived objects to be notified of the change so they can update their values accordingly. For example, when a property value of a prototypical object changes, an ObjectChangedEvent message is sent to all registered objects. Usually, this type of ObjectChangedEvent will contain the name of the changed property and the new value of the property. Consequently, when a derived object receives an ObjectChangedEvent, it will need to test the property name in the message against each property name that is of interest to the derived object until a match is found, at which point, the derived object's response to the notification can be carried out. This search process can be time-consuming if property changes on prototypical objects are frequent, but Berry does not discuss any techniques for optimizing this process.

By way of contrast, the present invention requires *a mapping data structure which stores computed mappings to an option binding*. When an object is notified of an option value change, the invention first searches the mapping data structure for the option binding, and if it was not previously computed, it then computes the mapping to the option binding and stores it in the mapping data structure. The precomputed mapping stored in the mapping data structure enables an option change to be effected seamlessly, without having to iterate through numerous case statements, or compare the names of all of its potential options to determine which option is affected by the change. As such, Berry does not relate to the claimed mapping data structure, as set forth in Claims 1, 10, 19, 20 and 22.

Moreover, Berry is silent as to how memory is allocated. It appears that Berry is consistent with traditional object-oriented techniques, as Berry does not suggest modifying them in any way. Therefore, in Berry, memory space is presumably allocated for each value, similar

to object-oriented languages, regardless of whether the value has been set on the object. As a result, memory is allocated for all the potential properties of an object, even if the properties are never used by the object.

The Examiner states that in Berry, only attribute values contained in the derived object need memory preallocation, while values that are set in the prototypical object do not require preallocation in the derived object. It is respectfully submitted, however, that in Berry, each attribute value that can be set on an object has a preallocated field in that object that will be set to the attribute value if a value is assigned for that attribute on that object. Even when this field is not set, it still occupies its own dedicated memory space in the object. It is true that some types of attribute values (such as variable-length character strings) may require additional allocation when they are set, but even in those cases, the object upon which the attribute value can be set must have a preallocated field (typically 4 bytes) to contain the address of the additional storage allocated for the attribute value. Berry does not discuss or disclose any technique for avoiding the allocation of this basic field, which is plainly shown as the field "fBackground" in Berry's coding example at col. 5, ll. 1-12. Even if the amount of preallocated storage per object is only 4 bytes for each attribute value that could be set on the object, storage space considerations can still become significant when a large number of attribute values can potentially be set on an object, but in practice, only a small number of attribute values usually are set on the object.

For example, when designing a graphical user interface, it can be desirable to have a vast number of style and behavior properties that can apply to groups of objects in the display. In a system of the Berry type, storage space would be allocated for each of the potential properties. Each of the potential property values would be allocated storage space during the life of the object, and if none of those values were actually set on the object, the storage may be considered wasted.

The present invention, however, enables an object to be defined with an option data structure that does not allocate storage space for full option values. Storage space is allocated when the option value is actually set on the object. In this way, the invention addresses memory

issues that are not contemplated by Berry. Accordingly, Berry does not relate to the claimed option data structure that supports references to nonlocal option values without preallocation of memory space for the full option values, as set forth in Claims 1, 10, 19, 20 and 22.

As such, it is respectfully submitted that the Examiner has not made a prima facie case under 35 U.S.C. § 102(b), because Berry does not teach every aspect of the claimed invention, namely:

- defining an object with an option data structure which supports references to option values without preallocation of memory space for the full option values; and
- notifying objects of a change in an option value through a change handler identified by an option binding, the option binding being located by first searching a mapping data structure for a previously computed mapping to the option binding and, if no mapping was previously computed, by then computing the mapping to the option binding and storing the mapping in the mapping data structure, as set forth in Claims 1, 10, 19, 20 and 22, respectively

Therefore, it is respectfully requested that the rejection of Claims 1, 10, 19, 20 and 22, and their respective dependent claims, under 35 U.S.C. § 102(b) be withdrawn.

Rejections under 35 U.S.C. § 103(a)

Claims 2, 5-8, 11, 14-17 and 21 were rejected under 35 U.S.C. § 103(a) based on Berry in view of Hostetter et al., "Curl: A Gentle Slop Language for the web," World Wide Web Journal, Spring, 1997. These rejections are traversed. For the reasons set forth above, Claims 1, 10, 19, 20 and 22 and their respective dependent claims are in condition for allowance. Reconsideration of the rejections under 35 U.S.C. § 103(a) is respectfully requested.

the present amendment. In [incr Tk], there is preallocation of memory space for option values and the application has been amended to correct any reference to the contrary. Thus, the Specification has been amended to correct a drafting error. Acceptance is respectfully requested.

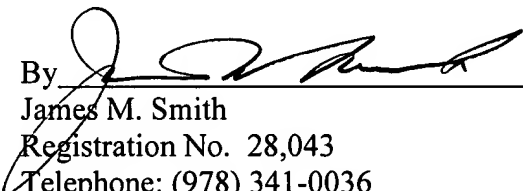
The Examiner also objected to the Abstract because it exceeded the 150 word limitation. The Abstract is amended to meet that limitation. No new matter is introduced. Acceptance is respectfully requested.

CONCLUSION

In view of the above amendments and remarks, it is believed that all claims are in condition for allowance, and it is respectfully requested that the application be passed to issue. If the Examiner feels that a telephone conference would expedite prosecution of this case, the Examiner is invited to call the undersigned attorney.

Respectfully submitted,

HAMILTON, BROOK, SMITH & REYNOLDS, P.C.

By 
James M. Smith
Registration No. 28,043
Telephone: (978) 341-0036
Facsimile: (978) 341-0136

Concord, MA 01742-9133
Dated: 